WEST

Generate Collection Print

L7: Entry 22 of 26 File: USPT Jul 3, 2001

DOCUMENT-IDENTIFIER: US 6256634 B1
TITLE: Method and system for purging tombstones for deleted data items in a replicated database

Abstract Text (1):
A method and system coordinates the purging of tombstones for data items deleted from a directory service database of a message queuing system. The directory service database is a replicated database with a plurality of servers, and the data items owned by one server are replicated by all other servers in the system. The directory service database supports a synchronization mechanism for a slave server to request for replication data from a master server. To support the synchronization mechanism, a tombstone is set up each time a server deletes a data item from its local database. For a slave server of a first type, the master server purges the tombstone after receiving an acknowledgment from the slave server for receipt of replication information regarding the deletion. For a slave server of a second type, the master server purges the tombstone after the tombstone has become sufficiently aged. If the slave server of the second type fails to receive the replication data and makes a synchronization request for a range of write operations including the deletion and the master server has already purged the tombstone for the deleted item, a full synchronization is performed between the master and slave servers such that the slave server reconstructs a fresh copy of data items currently in the database of the master server.

Brief Summary Text (2):
This invention relates generally to a replicated database, and more particularly to the replication of data items in a replicated database, such as one used in a message queuing system for providing directory service.

Brief Summary Text (5):
In a proposed message queuing system, a replicated database is maintained for providing directory service for message queuing and routing operations. This directory service database includes a plurality of local databases maintained by respective directory servers on different machines. Each directory server maintains not only data items created by itself but also a replica of data items created by all other servers in the directory service database. When a server creates, modifies, or deletes its data items, it sends replication message packets to the other servers so that they can update their respective replicas. In the context of data replication, a server sending replication information is referred to as a "master," and a server receiving the replication information is referred to as a "slave." If a slave server learns or suspects that it has not received all of the replication information from a master server, it sends a synchronization request to the master server to obtain the missing replication information. To support the synchronization operation, when a server deletes a data item from its local database, it sets up a tombstone to memorialize the deletion so that the deletion can be replicated by other servers. As the configuration of the message queuing system changes over time, data items representing message queuing objects, such as message queues, are constantly created and deleted. With more and more data items deleted from the directory service, the number of tombstones increases correspondingly. Without an effective way to purge the tombstones, the tombstones may grow in an unbounded way and ultimately may fill up the memory space of the directory service database. On the other hand, if the tombstones are purged prematurely, i.e., before other servers have learned about the deletion, the synchronization operation cannot be performed properly.

Brief Summary Text (7):
In accordance with the present invention, there is provided a method and system for purging tombstones for deleted data items in a replicated database in which data items owned by one server are replicated by other servers. When a master server in the

replicated database deletes a data item from its local database, it sets up a tombstone for the deleted data item and sends replication data regarding the deletion to one or more slave servers each of which maintains a replica of the master server's data items. The timing for purging the tombstone by the master server depends on the types of the slave servers receiving the replication data. For a slave server of a first type, the tombstone is purged only after the master server has received an acknowledgment from the slave server for receipt of the replication data regarding the deletion. For a slave server of a second type, the master server purges the tombstone after the tombstone has become sufficiently aged, without requiring the slave server to acknowledge that it has received the replication data regarding the deletion. In the event that the slave server fails to receive the replication data regarding the deletion and later requests for a synchronization to update its replica when the master has already purged the tombstone for the deletion, the master server cooperates with the slave server to perform a full synchronization to completely reconstruct the slave server's replica of data items of the master server. Efficient full synchronization methods are provided to enable the slave server to continue to serve read requests for the data items being updated during the full synchronization process.

Drawing Description Text (4):
FIG. 3 is a schematic diagram showing servers in the replicated database connected for data replication and synchronization operations;

Drawing Description Text (7):
FIG. 6 is a schematic diagram showing a replication packet sent by a master server to a slave server for data replication;

Drawing Description Text (8):
FIG. 7 is a schematic diagram showing an acknowledgment packet for receipt of replication data;

Detailed Description Text (12):
The MQIS includes a plurality of directory servers distributed at different Sites of the MQ Enterprise 68. More particularly, each Site is provided with a directory server called "Primary Site Controller" (PSC). A PSC is responsible for creating and maintaining data items representing MQ objects within its Site, including machines and message queues in the Site. One of the PSCs in the MQ Enterprise also serves as a "Primary Enterprise Controller" (PEC) 69 for creating and maintaining data items representing Users, Connected Networks (CNs), Sites, Site Links, and the Enterprise 68. Besides the data items it has created for its Site, each PSC also maintains a replica of data items created by the PEC 69 and all other PSCs for their respective Sites. Each Site may also include one or more directory servers called "Backup Site Controllers" (BSCs). Each BSC maintains a copy of all data items maintained by the PSC of its Site, including the data items created by the PSC of its Site and the replica of data items created by the PEC and all other PSCs. Thus, the data items created and maintained by one MQIS server are replicated by all other servers in the Enterprise. In this respect, data replication in the MQIS has no "scope," i.e., there is no requirement that certain data are to be replicated only by selected servers. It will be appreciated, however, that the method of purging tombstones according to the invention may be implemented in a replicated database which supports scopes of data replication.

Detailed Description Text (13):
In a preferred embodiment, the MQIS is a "singlemaster" replicated database in the sense that each data item has a particular owner. The data items for computers and queues in a Site is owned by that Site, and data items such as Enterprise, Sites, Site links, CNs, and Users are owned by the PEC. Only the owner of a data item can create, modify or delete the master copy of that data item. This single-master concept should not be confused with the "master-slave" relationship between two servers in the context of data replication which will be described in greater detail below.

Detailed Description Text (16):
The spanning tree of the data replication in the MQIS is such that a PSC (or PEC) sends replication data regarding its own data items to all other PSCs and the BSCs in its Site, and each of the receiving PSCs forwards the replication data it has received to the BSCs in its Site. More particularly, to enable the other PSCs to update their replicas of its data items, a PSC (or the PEC) periodically sends replication message packets to other PSCs in the Enterprise 68 regarding write operations it has recently performed on its data items. The term "write operation" is used broadly herein to include any operation performed by the server that changes the database contents, such as creating, modifying, or deleting a data item. Each PSC also sends replication

packets regarding write operations performed by itself and all other PSCs to the BSC(s) in its Site. Because of the time delays for the replication information to propagate across the network, at any given time there may be discrepancies between the contents of the local databases maintained by the individual directory servers in the Enterprise.

Detailed Description Text (17):
The replication of write operations in the MQIS is now described by way of example in reference to FIGS. 3 and 4. For simplicity and clarity of description, the illustrated example focuses on data items for message queues, and only two PSCs and their associated BSCs are shown in FIG. 3 to illustrate the data replication process. It will be appreciated, however, the same replication process is used when more PSCs and BSCs and other types of message queuing objects are involved in the data replication. Also for simplicity and clarity of description, only the message queue data items created by the PSC 70 and the replicas thereof maintained by the other servers are illustrated in FIG. 3.

Detailed Description Text (19):
For managing the replication of data items throughout the MQ Enterprise, each write operation performed by a PSC server on its own data items is assigned a sequence number unique to that server for identifying the write operation. FIG. 4 shows an exemplary numbered sequence 80 of write operations performed by the PSC 70 that results in the message queue data items in the local database 76 of the PSC 70 as illustrated in FIG. 3. Three types of write operations are shown in the illustrated sequence: the "CREATE" operation creates a new data item, the "SET" operation modifies the values and/or properties in an existing data item, and the "DELETE" operation deletes a data item from the database. In the illustrated sequence in FIG. 4, message queue data items Q1 through Q7 are created at sequence numbers 1-5, 13 and 15, respectively, and the data items Q3 and Q4 are then deleted at sequence numbers 11 and 16. As a result, items Q1, Q2, Q5, Q6, and Q7 remain in the local database 76 of the PSC 70.

Detailed Description Text (21):
As described above, one of the write operations is the DELETE operation. Since a deleted item is removed from the local database of the server, the sequence number of the DELETE operation will be lost unless a "tombstone" is maintained to memorialize the deletion. The term "tombstone" as used herein means broadly any record maintained for keeping track of the deletion of a data item by a server. The importance of maintaining tombstones for deleted items for data replication purposes will be described in greater detail below.

Detailed Description Text (23):
To enable other servers to replicate the changes it has made, each PSC (or the PEC) sends replication information to other PSCs in the MQ Enterprise and BSCs in its own Site. In the context of data replication, a server which sends or forwards replication information to other servers is referred to as the "master," and a server receiving the replication information is referred to as the "slave." The replication information is included in one or more replication message packets sent from the master server to the slave server. In a preferred embodiment, the replication message packets are transmitted in direct sessions between the servers. To that end, every two PSCs in the MQIS have to be on a Connected Network, and the PSC of a Site have to be able to form a direct session with the BSCs in its Sites.

Detailed Description Text (24):
When a slave server receives a replication packet (or a synchronization reply described in greater detail below), it scans the replication data included in the packet. If it determines that the replication data are complete, it modifies its own replica according to the replication data in the packet and updates a variable to indicate the last sequence number of the particular owner that has been replicated by the slave server. If the slave server is a PSC, it forwards the replication data to the BSCs in its Site so that the BSCs can apply the changes to their replicas of the data items maintained by the PSC. In this replication process between the PSC and its BSC, the PSC is the "master" and the BSC is the "slave."

Detailed Description Text (25):
Instead of sending one replication packet each time it modifies its local database, a PSC sends replication packets to each of it slaves at fixed intervals, such as every 10 seconds across Sites (e.g., one PSC to another PSC) and every 2 seconds within its Site (i.e., a PSC to the BSCs in its Site). When a PSC performs a write operation on its own data or replicates a change to its replica of the data of another owner, it prepares in

the memory an update data structure which forms the building block of replication packets. The PSC keeps an update queue for each of its slave servers. The update queue consists of pointers to update data structures that are not yet replicated to the corresponding slave server. For each write operation performed on its own data, the PSC adds such a pointer to each of the update queues for the other PSCs and the update queues for its BSCs. For each replicated change to its replica of the data of another owner, the PSC adds a pointer in the update queues for its BSCs. When the time for sending replication data to a slave server comes, a master server checks the pointers in the update queue for that slave server to include the corresponding update data structures in one or more replication packets and sends the replication packets to the slave server.

Detailed Description Text (27):
The replication header 98 includes a ucFlush field which is a flag for indicating whether the receiver of the replication packets should forward the received packets to other servers immediately, and an UpdateCount field which contains the number of data items to be updated. The update sections 99 each contains replication data for updating one data item (which may have been created, modified, or deleted in the last replication interval). The update sections 99 are ordered by the sequence numbers of the data items. A bCommand field in each update section indicates the operation (CREATE, SET, or DELETE) applied to the data item. A guidObject field contains the GUID of the MQ object represented by the data item. A guidMasterId field identifies the owner of the data item. More specifically, for data belonging to a Site, the guidMasterId is the GUID of the Site. For data owned by the PEC, the guidMasterId is a specific number used to indicate that the owner is the PEC. A snSN field contains the sequence number of the write operation applied to the data item by the owner of the item. The update section also contains replication data regarding the properties and values of the data item. In the case of a CREATE operation, all properties and values of the created MQ object are sent. In the case of a SET operation, only the changed properties and values are sent. In the case of a DELETE operation, only the properties kept in the deletion table are sent.

Detailed Description Text (28):
Each update section also includes a snPrev field which contains the snSN of the update section before it. This field is used to indicate that there is no gap between the previous update section and the present update section. In a replication packet, the snSN values of the update sections are consecutive because each write operation by the data owner is included in the replication packet. In other words, the snPrev of an update section equals the snSN of the same update section minus one. It will be appreciated, however, that snSN values in a replication packet may not be consecutive if the data replication is implemented with scopes such that the replication data of some data items are not sent to certain slave servers. Moreover, when the same format of the update section is used in a synchronization reply as will be described below, the snSN values of the update sections may not be consecutive.

Detailed Description Text (29):
For coordinating the purging of tombstones, each slave PSC is required to acknowledge the receipt of replication packets from the master (PSC or PEC). In a preferred embodiment, a slave PSC sends an acknowledgment packet to the master when the sequence number of the master reaches a multiple of a pre-selected number, such as 256. An acknowledgment packet sent by a PSC, as illustrated in FIG. 7, includes a guidPSCMasterId identifying the master PSC server to which the acknowledgment is sent. A guidAckedMasterId identifies the owner of the data items the acknowledgment referred to. This field is needed in the case that the master is the PEC which also serves as the PSC for its Site, where it is necessary to identify whether the data belongs to the PEC or the Site of the PEC. A SEQNUM contains the largest sequence number of the particular owner that the slave PSC acknowledges to have received. As will be described in greater detail below, a PSC or (PEC) purges a tombstone for the deletion of its own data only after receiving acknowledgments from all other PSCs for receipt of replication data including the deletion. In contrast, the BSCs in the MQIS are not required to acknowledge receipt of replication information from their PSCs. Instead, a BSC is required only to send periodically, such as every 12 hours, an acknowledgment to the PSC of its Site indicating that it is "alive."

Detailed Description Text (31):
In a preferred embodiment, it is the responsibility of a slave server to ensure that it has received all replication data from a master server. When the slave server determines or suspects that it has not received all replication data from the master server, it makes a synchronization request to the master for the missing replication

data. A synchronization request is triggered when a slave server receives a Hello packet and finds that the snLSN in the Hello packet for a particular owner is greater than the largest sequence number of that owner it has received. Similarly, a synchronization request may be triggered when the slave server receives a replication packet and finds a gap between the largest sequence number of the data of a particular owner in its local replica and the snPrev field of the update in the packet for that owner. In another situation, when a slave server is rebooted after being off-line for a while, it does not know whether it has missed some replication packets sent by a master server. In each of these situations, the slave server sends a synchronization request to the master server (or multiple master servers after rebooting) specifying the data owner and a range of sequence numbers of the replication data that it considers missing. In the illustrated embodiment, a BSC may send a synchronization request to its PSC. A PSC may send synchronization requests to all the other PSCs or the PEC. After sending a synchronization request, the slave server schedules to check later (e.g., after 40 min.) if the gap in the replication data as specified in the request has been closed by receiving a synchronization reply from the master. If the gap has not been closed, the slave sends a new synchronization request. The slave continues to monitor the gap and, if necessary, sends a new synchronization request until the gap is closed.

Detailed Description Text (35):
As more and more tombstones are set up in the MQIS, however, they take up a significant portion of the available memory space of the replicated database. For each deleted item, there are multiple tombstones set up for its deletion. This is because each server having a copy of that data item has to set up a tombstone when it deletes that copy. By way of example, referring to FIG. 3, when the PSC 70 deletes the data item Q3 (sequence number 11 in FIG. 4), it enters a deletion record (tombstone) in its deletion table 114. As will be described in greater detail below, the tombstone in the deletion table 114 for Q3 has already been purged by the PSC 70 and is therefore not shown in FIG. 3. Replication data for sequence numbers 6-13 (FIG. 4), including the deletion of Q3, are included in the replication packet 116 sent to the PSC 66. When the slave PSC 66 receives the packet 116, it deletes its copy of the item Q3 from its local database, sets up a S tombstone 118 in its deletion table 120, and forwards the replication data in a packet 117 to the BSC 67 in its Site. When the BSC 67 receives the packet 117, it deletes its copy of the item Q3 in its database and again sets up a tombstone 122 in its deletion table 124.

Detailed Description Text (37):
In accordance with the invention, a method and system for purging tombstones is provided that strikes a balance between the need to purge tombstones promptly to free up memory space and the need to maintain tombstones sufficiently long to support replication and synchronization operations. The timing for purging a tombstone by a master server from its database depends on the types of slave servers to which replication information regarding the deletion is sent. For a slave server of a first type, the master server deletes the tombstone only after the slave server has acknowledged the receipt of the replication data regarding the deletion. For a slave server of a second type, the master deletes a tombstone after the tombstone has become sufficiently aged, with the presumption that the slave server is highly likely to have received the replication data regarding the deletion. If the slave server fails to receive the replication data and later requests for a synchronization when the tombstone is already purged by the master server, the master server cooperates with the slave server to perform a "full synchronization" to reconstruct for the slave server a complete and up-to-date copy of data items of the particular owner maintained in the master's database. As part of the full synchronization process, data items already deleted from the master's database are also identified and deleted from the slave's database.

Detailed Description Text (39):
By way of example, in the illustrated embodiment of FIG. 3, the PSC server 70, which is the owner of the deleted data item Q3, purges the tombstone for Q3 only after it has received an acknowledgment for receiving replication data including the deletion of Q3 from each of the other PSCs in the MQ Enterprise, including the PSC 66. The acknowledgment of receipt of the replication data regarding a deletion may be explicit or implicit. For example, in the illustrated embodiment, the PSC server 70 receives an acknowledgment 126 from the PSC 66 for receipt of replication data up to a sequence number 13, which implies that the server 66 has received the replication data regarding the deletion, which has a sequence number 11.

Detailed Description Text (40):

In contrast, the PSC 70 does not wait for an acknowledgment regarding the deletion from the BSC 72 in its Site 74, and the BSC 72 is not required to send an acknowledgment about receiving the replication data. The PSC 70 purges the tombstone for Q3 if it has received acknowledgments from all other PSCs regarding that deletion and if it determines that the tombstone has become sufficiently old as indicated by its sequence number and its time stamp in the purge table. If the BSC 72 somehow fails to receive the replication information and later requests for synchronization for sequence numbers including the deletion of Q3 after the master PSC 70 has already purges the tombstone for Q3, the master server cooperates with the slave server to perform a full synchronization to construct a fresh copy of the data in the master PSC's database for the BSC.

Detailed Description Text (41):
This purging scheme provides efficient purging of tombstones while keeping the probability of the need for full synchronization low. The identification of the need for full synchronization and the performance of the full synchronization are automated. Moreover, as will be described in greater detail below, the servers are fully operational during the full synchronization and the cost of the full synchronization is reasonable. In the illustrated embodiment, the PSCs in the MQIS are likely to be connected by slow and expensive connections. It is therefore more costly to perform a full synchronization between two PSCS, which may require a transfer of a copy of all data items in the master's database to the slave. Since a master PSC purges a tombstone only after all other PSCs have acknowledged their awareness of the deletion, the costly fully synchronization between two PSCs is avoided. On the other hand, a PSC, being the main directory server and the owner of date items for the MQ objects in its Site, is expected to be in operation nearly at all times. A slave PSC should therefore be able to acknowledge the receipt of replication data without excessive delay.

Detailed Description Text (42):
In contrast, the connection between a PSC and a BSC in its Site is typically fast and less expensive, but the BSC may go off-line at random intervals. If an acknowledgment from each BSC in the Enterprise is required before purging a tombstone, one offline BSC may prevent all the other servers in the Enterprise from purging their tombstones. Thus, when the slave server is a BSC, the cost of delaying the purging of a tombstone indefinitely to wait for an acknowledgment is likely to outweigh the cost of the relatively rare event of performing a full synchronization operation.

Detailed Description Text (43):
For coordinating the purge operations by different servers in the system, each server (PEC, PSC or BSC) in the MQIS maintains a purge table: The purge table 128, as illustrated in FIG. 11, has multiple fields including snAcked, snAckedPEC, guidMasterId, snPurged, snMasterPurged, and SYNC_STATE. For each data owner (Site or PEC) in the MQ Enterprise, there is a corresponding entry in the purge table. The guidMasterId field in the entry contains the GUID of the data owner. If the entry is for a data owner other than the server maintaining the purge table, the snMasterPurged field contains the sequence number up to which the server maintaining the table knows that the owner has purged the tombstones, and the server maintaining the table is not allowed to purge beyond this number. This number prevents a slave from purging faster than its master. As described above, this number may be included in the snPurged field in the Hello packet sent out by a master server to its slave servers. The snPurged field contains the sequence number up to which the server maintaining the purge table (as opposed to the data owner) has purged tombstones for data items owned by the PSC (or PEC) to which the table entry pertains. The snAcked field is maintained by a PSC to keep track of acknowledgments from other (slave) PSCs regarding replication of its data. Similarly, the PEC (and only the PEC) maintains the columns of snAckedPEC to keep track of acknowledgments from the PSCs about data it owns as the PEC. The SYNC_STATE field, as will be described in greater detail below, is used by a slave server to indicate its status during a full synchronization operation with the master server to which the purge table entry pertains.

Detailed Description Text (49):
Another number, "snNewPurge," is then determined as the maximum sequence number of the records (tombstones) in the deletion table (FIG. 5) for the server's deleted data items that is less than or equal to the smaller of snPurge and snMinAcked and has a time stamp less than or equal to the current time minus T (step 138), where T is the pre-selected grace period, such as 7 days. The length of the grace period T is selected such that any on-line BSC is highly likely to receive the replication data regarding the deletion within the grace period, and any off-line BSC is highly likely to be back in operation and obtain the replication data regarding the deletion through regular

synchronization operations within the grace period. The server then deletes all records in the deletion table for its data that have sequence numbers less than or equal to snNewPurge (step 140). In effect, the server determines whether a tombstone should be purged according to whether the following expression is true:

Detailed Description Text (51):
In both cases (with the server purging the tombstones being the owner and not the owner) described above, if the server is a PSC, it checks whether its BSCs have sent in acknowledgments that they are alive within a pre-selected time-out period. If a BSC has failed to do so, the PSC issues an event log warning indicating the name of the BSC and that the BSC did not send an acknowledgment. The time-out period is preferably shorter than the grace period to give the BSC an opportunity to be put back on-line before the tombstone is purged so that the BSC may obtain the replication data through a regular synchronization operation instead of the more expensive full synchronization operation. The failure of the BSC to acknowledge, however, does not prevent the server from purging the tombstones.

Detailed Description Text (52):
Referring now to FIG. 13, when a master server (PEC or PSC) receives a regular synchronization request from a slave server (which may be a PSC or BSC) for data items of a particular owner (step 148), it checks the requested range of sequence numbers. If the master has not yet purged any tombstone in the requested range of sequence numbers, it puts replication data for the requested range in one or more Sync_Reply packets and sends the packets to the slave (step 152). On the other hand, if the requested sequence number range includes the sequence number (snPurged) up to which the master server has purged the tombstones (step 150), the master server will not be able to respond to the synchronization request because it no longer knows whether there are deletions before snPurged. For example, referring to FIG. 3, if the BSC server 72 fails to receive the replication packet 107 regarding sequence numbers 6-13 but later receives the replication packet regarding sequence numbers 14 to 17, it sends a synchronization request to the PSC 70 for replication data for sequence numbers 6 to 13. Since the PSC 70 has already purged the tombstone for the deletion of Q3 (sequence number 11), it will be unable to perform the regular synchronization.

Detailed Description Text (58):
The foregoing shows that the present invention provides an effective method and system for purging tombstones for deleted data items in a replicated database. For slave servers with slow connections to a master server, the master server purges a tombstone only after the slave servers have acknowledged receiving the replication data for the deletion. For slave servers with fast connections to the master but unpredictable downtimes, the master server purges the tombstone after the tombstone has sufficiently aged. In the event that a slave server requests for replication data and the master has already purged tombstones up to the requested range of data, the slave server initiates a full synchronization with the master to obtain a fresh copy of the data of the master. This approach allows efficient use of system resources by avoiding delay caused by waiting for acknowledges from servers with unpredictable downtimes while avoiding the cost of performing full synchronization over slaw connections.

Current US Original Classification (1):
707/100

Current US Cross Reference Classification (1):
707/101

Current US Cross Reference Classification (2):
707/202

Current US Cross Reference Classification (3):
707/206

CLAIMS:

1. A method of purging tombstones for memorializing deleted data items in a replicated database in which a slave server maintains a replica of data items stored by a master server, comprising the steps of:

performing a deletion by a master server to delete a data item stored by the master server;

setting up by the master server a tombstone for the deleted data item;

transmitting, from the master server to the slave server, a replication packet containing replication data regarding the deletion; and

purging, by the master server, the tombstone for the deleted data item after an age of the tombstone has reached a pre-selected threshold.

4. A method as in claim 1, further including the step of assigning an operation sequence number to the deletion of the data item for identification thereof, and wherein the replication data includes the operation sequence number of the deletion.

5. A method as in claim 1, further including the steps of:

transmitting, from the slave server to the master server, a synchronization request for information for updating the replica maintained by the slave server;

determining, by the master server, whether the synchronization request specifies an update range that covers the deletion of the data item and the tombstone for the deleted data item is purged by the master server; and

performing a full synchronization to reconstruct the replica of the slave server to match the data items stored by the master server when the update range specified in the synchronization request covers the deletion of the data item and the tombstone for the deleted data item is purged by the master server.

7. A method as in claim 6, wherein each data item has an operation sequence number associated therewith, and wherein the step of performing the full synchronization includes:

setting operation sequence numbers of data items of the replica maintained by the slave server to zero;

transmitting, from the master server to the slave server, replication data regarding the data items stored by the master server;

reconstructing the replica of the slave server according to the replication data regarding the data items stored by the master server; and

deleting, after the reconstructing, data items of the replica with a zero sequence.

10. The method of claim 1, further including the steps of:

transmitting replication data regarding the deletion to a second slave server which maintains a second replica of the data items of the master server;

awaiting, by the master server, an acknowledgment from the second slayer server of receipt of the replication data before purging the tombstone for the deleted data item.

11. A method of purging tombstones for memorializing deleted data items in a replicated database having first, second, and third servers each having a local database and the second and third servers each maintaining a replica of data items maintained by the first server, comprising the steps of:

performing a deletion by the first server to delete a data item from the local database of the first server;

setting up by the first server a first tombstone for the data item deleted by the first server;

transmitting, from the first server to the second server, a first replication packet containing replication data regarding the data item deleted by the first server;

transmitting, from the first server to the third server, a second replication packet containing replication data regarding the data item deleted by the first server;

receiving an acknowledgement from the second server to the first server regarding receipt of the first replication packet; and

purging, by the first server, the first tombstone after receiving the acknowledgement from the second server and after an age of the first tombstone has reached a pre-selected threshold.

13. A method as in claim 11, further including the steps of:

deleting, by the second server, the data item deleted by the first server from the replica maintained by the second server upon receiving the first replication packet;

setting up by the second server a second tombstone for deletion of the data item from the replica of the second server; and

sending, by the second server, a third replication packet containing replication data regarding the deleted data item to a fourth server of the replicated database, the fourth server maintaining a replica of data items maintained by the second server.

15. A method as in claim 11, further including the steps of:

transmitting, from the third server to the first server, a synchronization request for information for updating the replica of the third server;

determining, by the first server, whether the synchronization request specifies an update range that covers the deletion of the data item and the first tombstone is purged by the first server; and

performing a full synchronization to reconstruct the replica of the third server to match the data items of the first server when the update range specified in the synchronization request covers the deletion of the data item and the first tombstone is purged by the first server.

17. A replicated database system comprising:

a first server having a first database for storing data items, a deletion table for storing deletion records for data items deleted by the first server from the first database, and a purge table for storing data regarding purge operations performed by the first server on the deletion records in the deletion table, each deletion record in the deletion table having a time stamp for deletion of a corresponding data item, the purge table including a field indicating an extent to which the first server has purged the deletion records in the deletion table; and

a second server having a second database for maintaining a replica of the data items stored in the first database,

the first server programmed to place a deletion record in the deletion table for a data item deleted from the first database, send replication data regarding the deleted data item to the second server for updating the replica maintained by the second server, and purge the deletion record after the deletion record has reached a pre-selected age.

18. A replicated database system as in claim 17, further including a third server having a third database for maintaining a replica of the data items stored in the first database, and wherein the first server programmed to purge the deletion record for the deleted data item after receiving an acknowledgment from the third server for receiving replication data sent by the first server regarding the deletion of the deleted data item.

21. A computer-readable medium having computer-executable instructions for performing steps comprising:

performing a deletion by a master server in a replicated database to delete a data item stored by the master server;

setting up by the master server a tombstone for the deleted data item;

transmitting, from the master server to a slave server in the replicated database, a replication packet containing replication data regarding the deletion, the slave server maintaining a replica of the data items stored by the master server; and

purging, by the master server, the tombstone for the deleted data item after an age of

the tombstone has reached a pre-selected threshold.

22. A computer-readable medium having computer-executable instructions for performing steps comprising:

performing a deletion by a first server in a replicated database to delete a data item from a local database of the first server;

setting up by the first server a first tombstone for the data item deleted by the first server;

transmitting, from the first server to a second server in the replicated database, a first replication packet containing replication data regarding the data item deleted by the first server, the second server maintaining a replica of data items in the local database of the first server;

transmitting, from the first server to the third server in the replicated database, a second replication packet containing replication data regarding the data item deleted by the first server, the third server maintaining a replica of data items in the local database of the first server;

sending an acknowledgment from the second server to the first server regarding receipt of the first replication packet; and

purging, by the first server, the first tombstone after receiving the acknowledgment from the second server and after an age of the first tombstone has reached a pre-selected threshold.